Unsupervised Cross-Domain Image Generation

Girish Chandar G
IIT Gandhinagar
girish.chandar@iitgn.ac.in

S Deepak Narayanan
IIT Gandhinagar
deepak.narayanan@iitgn.ac.in

Sammed S Kagi
IIT Gandhinagar
sammed.shantinath@iitgn.ac.in

Shivji Bhagat
IIT Gandhinagar
shivji.bhagat@iitgn.ac.in

ABSTRACT

We address the problem of unsupervised general domain transfer by analysing and implementing a modified version of the method suggested in [1]. We use a modified GANs architecture to transfer image samples from one domain to another. Specifically, two different image domains were explored : Face and Digits. We were able to obtain qualitatively and quantitatively appealing results in case of digits transfer.

KEYWORDS

general domain transfer, Generative Adversarial Neural Networks (GANs), feature encoder

ACM Reference format:
Girish Chandar G, S Deepak Narayanan, Sammed S Kagi, and Shivji Bhagat. 2019. Unsupervised Cross-Domain Image Generation. In *Proceedings of IIT Gandhinagar's ML Course, Gandhinagar, India, Spring 2019 (Machine Learning),* 5 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

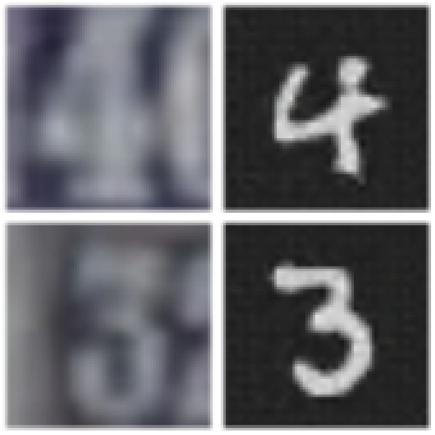


Figure 1: Domain Transfer Example

1 INTRODUCTION

Humans have inherently had the expertise in mapping different domains with very little supervision. This is something that machines have thus far not had great success at. To clarify on what a domain is, exactly we'd like to take this example of two different types of images here. The images on the first column are said to belong to one domain (printed numbers) while those on the second column are said to belong to another domain (handwritten numbers).

This work tries to progress towards the ultimate goal of Artificial General Intelligence by attempting to solve this non-trivial problem. One potential application we would have out of solving this problem will be a uniform digit recognizer. We would train a model which can recognize digits across various domains that is irrespective of the background and environment we can identify numbers . This would be very important in the current world with so many different variants of digits existing across even related domains.

General domain transfer refers to mapping of samples from one domain to a similar sample in another domain. One such example is transferring digit images from Street View House Number (SVHN) data set to MNIST[1] data set of handwritten digits. It is an important problem in the field of machine learning and has various practical applications.

To address this problem of generalized domain transfer we implement and analyse various modifications of the approach suggested in "Unsupervised Cross-Domain Image Generation" [1]. We specifically explore two areas - Digit transfer (SVHN to MNIST and vice versa) and Face Transfer (MS Celeb to Bitmozi). Thus, for any given sample in source domain, S we aim to transfer the sample to a similar target domain, T, in unsupervised fashion.
For this purpose , we use a modified architecture of GANs as proposed in [2]. A feature representation of the input samples is given as input to the GAN and it is trained with samples of target domains.

2 RELATED WORK

The proposed architecture that the authors give is actually inspired by the successful implementation of architecture by Radford et al's work where they propose DCGAN [3]. Another related work is by Dosvitiskiy et al.[4], which works on a similar problem has successfully mapped embedding to their pre-images, given source-target pairs. This particular work has a GAN as well as additional losses, which has in part inspired the authors to employ a similar loss function, with GAN and additional losses covering various other constraints. [5] was a seminal work in understanding image representations that are useful.

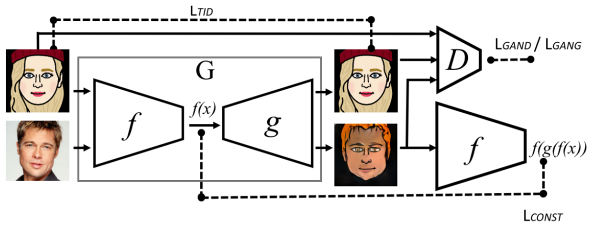


Figure 2: The proposed architecture

As the authors put it "As far as we know, the domain transfer problem we formulate is novel despite being ecological (i.e.,appearing naturally in the real-world), widely applicable, and related to cognitive reasoning (Fauconnier & Turner, 2003)." Therefore there is no prior state-of-the-art method apart from [1].

Yet another motivation for this work has been from various different work in the domain on Image Style Transfer. This line of research has focused on creating new images, while simultaneously minimizing content loss with respect to one domain and style loss with respect to the other. Here the word loss does not refer to the loss used for optimization, but rather to the actual loss in terms of content and style of the image. There are a lot of similarities between Style Transfer and the current work. Both are unsupervised and generate a sample under a given constancy constraint. Image style transfer tries to replicate the style of one or several images, while this work considers a distribution in the target space.

3 DATASETS

Application	Dataset	Type of Images
Digit Transfer	SVHN	RGB
	MNIST	Grayscale
Face Transfer	MS-Celeb	RGB
	Bitmoji	RGB

The SVHN(Street View House Numbers), MNIST and MS-Celeb datasets are readily available. The Bitmoji dataset needs to be generated by us. [4] is a re-implementation of the original work [1], and the authors in [4] have generated the dataset for Bitmoji using a specific method which is described in their report. We would firstly like to contact the original authors regarding the dataset and depending on their response, would generate them if needed, in the same manner, used for generation in [4].

4 APPROACH

There are two domains, namely, the source domain s and the target domain t. We need to generate samples in domain t corresponding to samples in domain s. The domains in our case are images.
The authors approached the problem of unsupervised cross-domain image transfer by a modified GAN architecture as shown in Figure 6.2. The block G in the Figure 6.2 consists of two blocks, namely, the feature encoder f and the generator g ,i.e, $G = gof$.

In a traditional GAN implementation the input to the generator in a random vector sampled from a prior distribution (eg. Gaussian distribution). But in the implementation proposed by the authors, the input to the generator is a feature vector from the feature encoder f.

The feature encoder f is a part of a classifier pre-trained on the source domain s. In general the feature encoder is the classifier with the last fully-connected layer removed. Thus the feature encoder outputs an n-dimensional feature vector which then would be given as an input to the generator. For $x \in s, t$ $f(x)$ outputs the features that best defines x.

Thus for $x \in s, t$ $G(x) = g(f(x))$ outputs an image generated by the generator block g with feature vector $f(x)$ as input.

The discriminator D performs a ternary classification which the authors claim to be robust as opposed to the discriminator of traditional GANs with binary classification. The discriminator D classifies its input image into one of the three classes: 1) Image from source domain passed through generator, 2) Image from target domain passed through the discriminator, 3) Image from the target domain.

The modified GAN implementation proposed by the authors employs additional loss functions to the generator apart from the traditional GAN loss to impose extra constraints. The loss functions associated with the generator are as follows:

$$L_{GANG} = -\sum_{x \in \mathbf{s}} \log D_3(g(f(x))) - \sum_{x \in \mathbf{t}} \log D_3(g(f(x)))$$

$$L_{CONST} = \sum_{x \in \mathbf{s}} d_1(f(x), f(g(f(x))))$$

$$L_{TID} = \sum_{x \in \mathbf{t}} d_2(x, g(f(x)))$$

$$L_{TIV} = \sum_{i,j} (z_{i,j+1} - z_{i,j})^2 + (z_{i+1,j} - z_{i,j})^2$$

- L_{GANG} is the ordinary GAN loss where generator tries to generator tries to create fake data and the discriminator tries to decide whether its real or fake.
- L_{CONST} helps in reducing the distance between the image from the source domain and the generated image from the source image in the feature space defined the feature encoder, i.e., the L_{CONST} makes sure that the features of the source domain image and the generated image are similar. The

distance metric d_1 is the l_2 norm and cosine similarity for digit transfer and face transfer respectively.

- L_{TID} enforces the identity mapping for images sampled from the target domain. It ensures that for $x \in s$, $g(f(x))$ generates the image in the target domain t corresponding to to x. The distance metric d_2 is the l_2 norm for both digit and face transfer.
- L_{TIV} is a smoothing loss enforced only for face transfer. This loss improves the aesthetics of the image generated.

The generator loss L_G is the sum of L_{GANG} and the weighted sum of L_{CONST}, L_{TID} and L_{TIV} with weights α, β, γ respectively. In the case of digit transfer γ is set to zero as we we do not impose the smoothing loss in digit transfer. Therefore the final generator loss is as follows:

$$L_G = L_{GANG} + \alpha L_{CONST} + \beta L_{TID} + \gamma L_{TIV}$$

The discriminator loss is sum of its ternary classification losses and it is formulated as follows:

$$L_D = -\sum_{x \in \mathbf{s}} \log D_1(g(f(x))) - \sum_{x \in \mathbf{t}} \log D_2(g(f(x))) - \sum_{x \in \mathbf{t}} \log D_3(x)$$

Adam optimizer [6] is used as the optimization algorithm for the entirety of the implemenetation

5 EXPERIMENTAL SETUP

5.1 Digit Transfer

As stated in the approach, the generator takes the feature encoded inputs of the source domain samples and the target domain samples. We trained our f starting from training a classifier of digits on the SVHN dataset. The network structure of the classifier (f) consists of four blocks of convolutional layers and a ReLU nonlinearity layer. For the first three convolutional layers we used kernel size of 2 , padding of 1 and average pooling to shrink the size of the image, with the number of filters equal to 64, 128, 256 respectively. For the last layer, we use a kernel size of 4 with padding of 0 and number of filters equal to 128. This architecture gives us a 128 x 1 x 1 vector as a output. This is sent through a fully-connected layer to shrink it to the output size of 10, the number of labels for the digits).

This classifier is then trained on the SVHN images. We obtain an accuracy of **94%**. We take the first 10 layers out of the 15 layers as the f block in our digit model, so that is encodes the features from the image. The output of this segment is a 128 x 1 x 1 vector.

The generator g maps f's feature vector to a 32 x 32 grayscale image. g employs four blocks of transposed convolution, batch-normalization and ReLU. The first layer has a kernel size of 4, stride of 1, padding of 0, with the number of filters being equal to 512. The remaining 3 layers have a kernel size of 4, stride of 2, padding of 1 with number of filters 256, 128, 1 respectively. We add a *Tanh* layer at the end to normalize g's output between [-1,1].

The Discriminator D takes in the 32x32 image from the generator and returns 3 probabilities of the image belonging to each of the three classes. D employs 4 blocks of convoloution followed by batch normalization and Leaky ReLU. The first three layers have kernel size 3, stride 2 , padding 1 and number of filters equal to 128, 256, 512 respectively and the last layer has a kernel size 4 and stride 2 and number of filters equal to 3.

5.2 Face Transfer

In the original paper, the network for face image transfer builds upon the representation layer of the DeepFace [7] network for the important f block, which is not publicly available. We were therefore left to find a viable alternative to use for f. DeepFace is trained by Facebook in house. The kind of scale and dataset that Facebook has trained is impossible for us to do. This naturally made our face transfer a much more harder problem to implement and train. We were left with no option but to resort to open sourced implementations of good facial feature encoders. We chose to use SphereFace[8] as our representation layer. SphereFace was the more viable open-sourced alternative that we had that was pre-trained. The generator g and discriminator D were implemented following the original paper. g takes in the feature vector from f and outputs a 64 x 64 RGB image. This image that we produce is upsamples using bilinear interpolation. The interpolated image is finally upsampled to a size of 96 x 96 and g containes 5 blocks, each containing stride 2, padding 1, kernel size 4 transposed convolution with number of filters 512, 256, 128, 64, 32 respectively followed by batch normalization and a ReLU. Another 1 x 1 convolution was added with stride 1, padding 0 and kernel size 1 after each block in an attempt to lower final L_{CONST} values. After these 5 blocks, a final transposed convolution is performed with kernel size 4, stride 2 and padding 1, followed by a Tanh output to normalize output between [-1,1].

Discriminator D similarly contains 5 blocks, each of which contain a stride 2, padding 1, kernel size 3 convolution followed by batch noralizaton and Leaky ReLU non-linearity with $\alpha = 0.2$. The final output is a convolution with 3 filters kernel size 3 stride 1. The number of filters of the convolutional layers varies, but generally the first convolution has few filters (32,62,128), which increased to 4 times more in the middle, then decreases to reach 3 in the output.

6 RESULTS

We successfully implemented the proposed Domain Transfer Network in Google Cloud with 8 Intel CPUs and 1 NVIDIA Tesla P100 GPU.

6.1 Digit Transfer

6.1.1 SVHN to MNIST. We were successfully able to implement the domain transfer network for SVHN-MNIST and our model output images gave an accuracy of 77.6% on MNIST classifier. The authors reported an accuracy of 90% in their paper. The final values for weights of the loss functions of our model are $\alpha = 0.001$ $\beta = 1000$ $\gamma = 0$. The low value of α can be attributed to the performance of the feature encoder. Since our SVHN classifier showed an accuracy of 94%, the feature encoder was able to extract features that define the image well.

6.1.2 MNIST to SVHN. The implementation of the domain transfer network for MNIST-SVHN and our model output is displayed

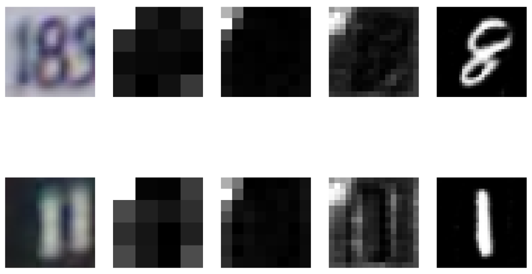


Figure 3: SVHN to MNIST - Output Image of each layer of the generator

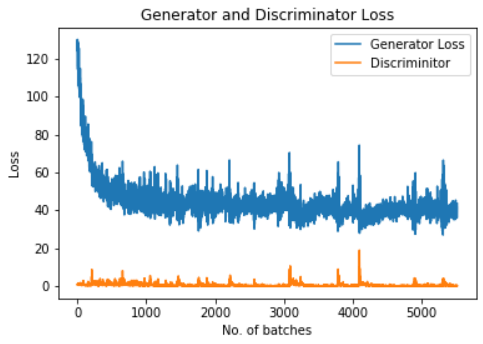


Figure 4: Digit Transfer Generator and Discriminator Loss

below. We trained an MNIST classifier with **98.1%**. We had to perform the same domain transfer, except for the following changes. We have a new classifier, since the target and source domains have now swapped. The up sampling that is performed using transposed convolution in the generator now has to end with three channels, since we are generating a RGB image from a grayscale image. Also, the tradeoff hyperparameters are $\alpha = 0.001$, $\beta = 1000$, $\gamma = 0$.

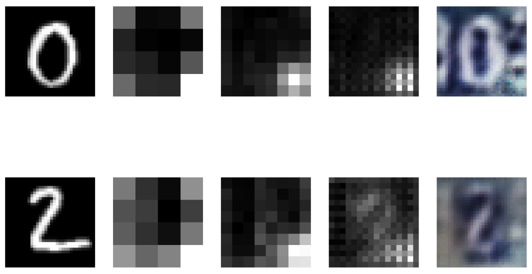


Figure 5: MNIST to SVHN - Output Image of each layer of the generator

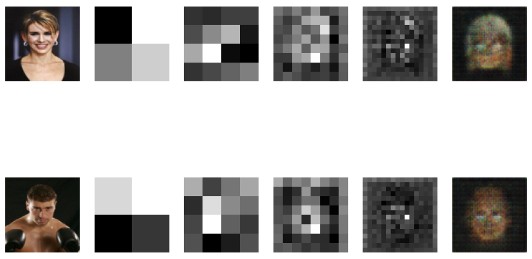


Figure 6: Face to Bitmoji - Output Image of each layer of the generator

6.2 Face Transfer

The face transfer model that we implemented did not turn out to be very successful. We were very often ending up in mode collapse situation, with absolutely no resemblance of faces in some extreme cases. Here we display the generation of a face from an input image. The trade-off hyper-parameters α= 0.01, β = 100, γ = 0.0001.

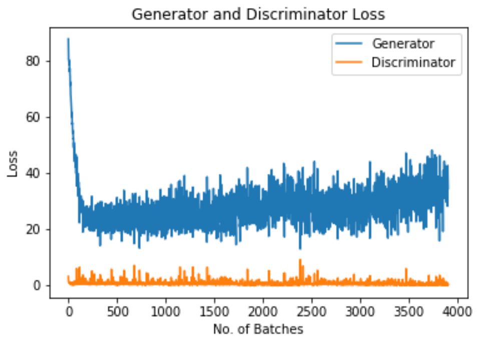
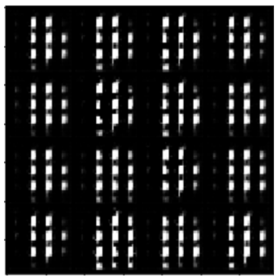


Figure 7: Face Transfer Generator and Discriminator Loss

7 DISCUSSION AND CONCLUSION

- The digit transfer was successful but the face transfer did not give satisfactory results.
  - This problem might have been due to the reduced complexity of the feature encoder of the faces since capturing features of digits are far more easier than faces.
  - The authors at Facebook AI Research had access to their proprietary classifier which had been train their classifier on 4 million images and thus their face encoder was able to capture the features of the face images very well.
  - In the future, by improving the face feature encoder, we might be able to get better results.
- The image transfer is bi-directional, i.e., the domain transfer network works with source and target domain swapped. We have shown this conclusively by performing digit transfer from SVHN to MNIST and also from MNIST to SVHN.

- Training a GAN was a very difficult process. Several times we encountered mode collapse. During mode collapse GAN rests in an local optima and thus the generator outputs the same image independent of the input image. Figure 8 shows a mode collapse example for digit transfer (8(a)) and face transfer (8(b))
- Our entire code will be made available by May 4th 2019 on GitHub, and we will share the link at that time.


(a) Digit Transfer

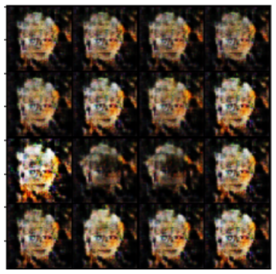

(b) Face Transfer

Figure 8: Mode Collapse

REFERENCES

[1] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[2] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[3] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[4] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in neural information processing systems*, pages 658–666, 2016.

[5] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[8] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.